Claims

[c1] 1.A method for creating a logic design using a register transfer language, said method comprising:

upon a declaration of a signal interconnect, defining a language extension of a register transfer language for the signal interconnect based on the signal interconnect"s type;

storing different signal interconnect types in a same design file; and

creating a logic design by partitioning between different types of cells in the logic design, wherein the different types of cells are based on the different signal interconnect types as described by the language extensions.

[c2] 2.The method of claim 1, wherein the same design file is a register transfer language (RTL) design file.

[c3] 3.The method of claim 1, wherein the signal interconnect type is either a field programmable gate array (FPGA) type interconnect or a standard cell type interconnect.

[c4] 4.The method of claim 3, further comprising:

reading the same file with a tool that, based on available interconnect types, extracts an FPGA portion of the logic design and a standard cell portion of the logic design, wherein each portion of the logic design is synthesized using a synthesis tool appropriate for the type of interconnect.

[c5] 5.The method of claim 3, further comprising:
dynamically changing the language extension upon a change in the signal interconnect type.

[c6] 6.The method of claim 5, wherein the step of dynamically changing the language extension includes first changing the language extension to an intermediate language extension, which is capable of being accepted or rejected, to reflect a suggested change in the signal interconnect type, and upon the suggested change in the signal interconnect type being accepted, changing the intermediate language extension to a final language extension that describes the accepted changed signal interconnect type.

[c7] 7.The method of claim 6, further comprising:
repeatedly changing the signal interconnect type from an FPGA type to a standard cell type until a minimum die size is achieved.

[c8] 8.The method of claim 6, further comprising:
repeated changing the signal interconnect type from a standard cell type to an FPGA type to make a more flexible logic.

[c9] 9.A system for creating a logic design using a register transfer language, said system comprising:
means for, upon a declaration of a signal interconnect,

defining a language extension of a register transfer language for the signal interconnect based on the signal interconnect"s type;

means for storing different signal interconnect types in a same design file; and

means for creating a logic design by partitioning between different types of cells in the logic design, wherein the different types of cells are based on the different signal interconnect types as described by the language extensions.

[c10] 10.The system of claim 9, wherein the same design file is a register transfer language (RTL) design file.

[c11] 11.The system of claim 10, wherein the signal interconnect type is either a field programmable gate array (FPGA) type interconnect or a standard cell type interconnect.

[c12] 12.The system of claim 11, further comprising:

means for dynamically changing the language extension upon a change in the signal interconnect type.

[c13] 13.The system of claim 12, wherein the means for dynamically changing the language extension includes means for first changing the language extension to an intermediate language extension, which is capable of being accepted or rejected, to reflect a suggested change in the signal interconnect type, and upon the suggested change in the signal interconnect type

being accepted, changing the intermediate language extension to a final language extension that describes the accepted changed signal interconnect type.

[c14] 14.The system of claim 13, further comprising:
means for repeatedly changing the signal interconnect type from an FPGA type to a standard cell type until a minimum die size is achieved.

[c15] 15.The system of claim 13, further comprising:
means for repeatedly changing the signal interconnect type from a standard cell type to an FPGA type to make a logic more flexible.

[c16] 16.A computer program product, residing on a computer usable medium, for creating a logic design using a register transfer language, said computer program product comprising:
program code for, upon a declaration of a signal interconnect, defining a language extension of a register transfer language for the signal interconnect based on the signal interconnect"s type;
program code for storing different signal interconnect types in a same register transfer language (RTL) file; and
program code for creating a logic design by partitioning between different types of cells in the logic design, wherein the different types of cells are based on the different signal interconnect types as described by the language extensions.

[c17] 17.The computer program product of claim 16, wherein the signal interconnect type is either a field programmable gate array (FPGA) type interconnect or a standard cell type interconnect.

[c18] 18.The computer program product of claim 17, further comprising:
program code for dynamically changing the language extension upon a change in the signal interconnect type.

[c19] 19.The computer program product of claim 18, wherein the program code for dynamically changing the language extension includes program code for first changing the language extension to an intermediate language extension, which is capable of being accepted or rejected, to reflect a suggested change in the signal interconnect type, and upon the suggested change in the signal interconnect type being accepted, changing the intermediate language extension to a final language extension that describes the accepted changed signal interconnect type.

[c20] 20.The computer program product of claim 19, further comprising:
program code for repeatedly changing the signal interconnect type from an FPGA type to a standard cell type until a minimum die size is achieved.

[c21] 21.A single placeable logic tile comprising:

a first portion containing a field programmable gate array (FPGA) logic; and

a second portion containing a standard cell logic, wherein the first portion and the second portion are synthesized from a single register transfer language (RTL) file having both FPGA and standard cell files.

[c22] 22.An array of placeable logic tiles, said array comprising:

a plurality of placeable logic tiles, each of said plurality of placeable logic tiles having a same outline size but a different amount of field programmable gate array (FPGA) and standard cell logic, wherein a first placeable logic tile from said plurality can replace a second placeable logic tile from said plurality to effectively move a boundary between the FPGA and standard cell logic within the tile.

[c23] 23.A set of placeable logic tiles comprising:

a plurality of placeable logic tiles, each of said plurality of placeable logic tiles having a different outline size and a different amount of field programmable gate array (FPGA) and standard cell logic, wherein a first placeable logic tile can replace a second placeable logic tile of different outline size.